Journal of Network and Systems Management, Vol. 7, No. 4, 1999

Rule Discovery in Telecommunication Alarm Data

Mika Klemettinen,[1] Heikki Mannila,[1,2] and Hannu Toivonen[1,3,4]

Fault management is an important but difficult area of telecommunication network management: networks produce large amounts of alarm information which must be analyzed and interpreted before faults can be located. So called alarm correlation is a central technique in fault identification. While the use of alarm correlation systems is quite popular and methods for expressing the correlations are maturing, acquiring all the knowledge necessary for constructing an alarm correlation system for a network and its elements is difficult. We describe a novel partial solution to the task of knowledge acquisition for correlation systems. We present a method and a tool for the discovery of recurrent patterns of alarms in databases; these patterns, episode rules, can be used in the construction of real-time alarm correlation systems. We also present tools with which network management experts can browse the large amounts of rules produced.

The construction of correlation systems becomes easier with these tools, as the episode rules provide a wealth of statistical information about recurrent phenomena in the alarm stream. This methodology has been implemented in a research system called TASA, which is used by several telecommunication operators. We briefly discuss experiences in the use of TASA.

KEY WORDS: Alarm correlation; fault identification; rule discovery; data mining; episodes.

1. INTRODUCTION

Telecommunication networks are growing fast in size and complexity, and at the same time their management is becoming more difficult. The task of identifying and correcting faults in telecommunication networks is a critical task of network management: faults that interfere with the services offered by the network are

[1] Department of Computer Science, University of Helsinki, Finland.
[2] Current affiliation: Nokia Research Center, Finland.
[3] Current affiliation: Rolf Nevanlinna Institute, University of Helsinki, Finland.
[4] To whom correspondence should be addressed at Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland. E-mail: *Hannu.Toivonen@cs.helsinki.fi*

1064-7570/99/1200-0395$16.00 /0

costly for the operator. The quality of services plays also an important role in the growing competition between operators.

Network elements produce large amounts of alarms about the faults in a network. Fully employing this valuable data in network management is difficult, however, due to the high volume and the fragmented nature of the information. Moreover, changes in equipment, software, and network load mean that the characteristics of the alarm data change.

Our approach to the task of processing alarms is supplementary to alarm correlation, a central technique in fault identification [1–8]. In alarm correlation, a management center automatically analyzes the stream of alarms, notifications, and clear messages it receives from a telecommunication network. Alarm correlation is typically based on looking at the active alarms within a time window, and interpreting them as a group. This interpretation can result in filtering of redundant alarms, identification of faults, and in suggestions for corrective actions. The goal of alarm correlation systems is to reduce the workload of network managers by processing the large alarm data set into a smaller but more useful set of reports.

While the use of alarm correlation techniques is popular and methods for expressing the correlations are maturing, acquiring all the knowledge necessary for constructing an alarm correlation system for a network and its elements is difficult. The complexity and diversity of network elements and the large variation in the patterns of alarm occurrences pose serious problems for network management experts building a correlation model.

In this paper, we present methods for semi-automatic discovery of patterns in alarm databases; these methods help in the construction of alarm correlation systems. We use novel algorithms to discover recurrent patterns in large alarm databases. We then apply iterative information retrieval methods to give flexible views to the discovered patterns. The process we propose for building alarm correlation systems contains the following three steps:

1. Semi-automatic discovery of alarm patterns (off-line).
2. Construction or modification of an alarm correlation system. The expert knowledge has a key role here, and the purpose of the discovered patterns is only to aid the experts in recalling and formulating correlation patterns.
3. Application of the correlation system in real-time alarm management.

We contribute to the first step of the process, the semi-automatic discovery of recurrent patterns of alarms. Steps 2 and 3 have been discussed elsewhere [1–8].

The ideas expressed in this article have been implemented in a system called TASA, for Telecommunication Alarm Sequence Analyzer. The TASA system has been developed in co-operation with four telecommunication compa-

nies. These companies have been using a prototype version of TASA with good results.

The rest of this article is organized as follows. In Section 2 we briefly review network surveillance and alarm correlation, and describe our scenario for building alarm correlation systems with the help of TASA. Our contribution starts from Section 3 where we present our rule formalisms and outline the algorithm. We then describe in Section 4 the whole pattern discovery process using the TASA system. Experiences in using the methods and performance results are discussed in Section 5. Section 6 is a short conclusion.

1.1. Related Work

For related work in the area of alarm correlation see [1–8]. Similar approaches have been used successfully also in process control tasks [9]. For a recent survey about fault management in communication networks, see [10].

Our approach and methods draw from the field of Knowledge Discovery and Data Mining (KDD); for overviews, see [11, 12]. KDD can be loosely defined as the task of obtaining useful and interesting knowledge from large collections of data. It combines methods and tools from machine learning, statistics, and databases. A related approach for the automatic acquisition of network management knowledge from the existing data has been considered before in [13].

Our approach of discovering all frequent patterns can be contrasted with numerous methods, e.g., in machine learning, which are more focused and produce one or at most a few patterns that match the given problem specification. These methods usually require one target concept, e.g., one particular fault, and they find regularities related to this concept but leave any other potentially interesting phenomena unfound. The advantage of those systems is that the patterns they find are more expressive than the relatively simple rules that we use.

Something can also be said of various KDD systems. *Explora* [14, 15] finds interesting instances of statistical patterns. The patterns discovered by *49er* [16] are contingency tables, equations, and logical equivalences. The Key Finding Reporter (Kefir) [17, 18] tailored with a lot of domain knowledge, discovers and explains deviations, and gives recommendations for corrective actions. To the best of our knowledge, none of these systems is directly applicable to discovery in temporal sequences of events such as alarms. These systems also differ substantially from the methodology we propose. Until recently, most knowledge discovery methods were designed for the analysis of unordered collections of data. The interest in sequential data has now increased: see e.g., [19–26].

Our algorithm for the discovery of frequent episodes works in a generate-and-test manner. The testing phase is similar to matching correlation rules to the stream of incoming alarms (see e.g., the Rete algorithm [27] used by a number of expert systems). There are similarities also to string matching (e.g., [28]).

Matching episodes to a sequence can be seen as locating all occurrences of subsequences, or matches of patterns with variable length "don't care" symbols, where the length of the occurrences is limited by the window width. Recent results on the pattern matching aspects have been reported by Das *et al.* [29].

2. ALARM CORRELATION

Alarm correlation is a central technique for processing the flow of alarms arriving in a management center into a smaller but more useful set of reports. In this section we give a brief overview of the task of correlating alarms.

2.1. Alarms

Faults in a telecommunication network are reported to management centers in the form of alarms. An *alarm* is a message emitted by a network element, typically when a problem is encountered. Unfortunately, a network element has a very narrow view to the network, and can therefore only report the symptoms of the fault from its limited viewpoint. On the other hand, one fault can result in a number of different alarms from several network elements.

We view an occurrence a of an alarm as a triple $a = (t, s, m)$, where t is the *time* of the alarm a, s is the *sender* of the alarm, and m is the *alarm message*. The time is recorded by the sender, typically at a granularity of one second. The sender of the alarm can be identified at the level of, e.g., a network element or a managed object. The alarm message describes the problem with the information that is available to the sender.

In addition to alarms, different notifications and clear messages are emitted by network elements. There are also events originating from the network management personnel, such as acknowledgments of alarms. All these different types of messages should be handled in alarm correlation systems. Although we for simplicity only talk about alarms in the rest of this article, the methods and ideas should be understood to cover any types of events and messages available.

The information contents of alarm messages vary a lot. Some alarms concern problems in logical concepts, such as virtual paths, some concern physical devices, such as power supplies. Some alarms report a distinct failure, e.g., that the incoming signal is missing, whereas some only report a high error rate without any hint for the cause.

Example 1. An example of the format of an actual alarm is given in Fig. 1. Here, the *date* and *time* fields make up the alarm time t and the *alarming element* field is the sender s. The alarm message m is then considered to contain the information of the rest of the fields. Actually, the alarm type often determines uniquely the other information of m, and in such cases it is possible to consider the alarm type alone as the alarm message m.

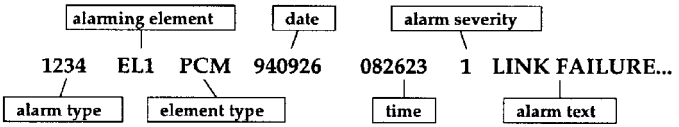

Fig. 1. An example of the format of an actual alarm.

2.2. Correlation

The task of a management center is to correlate the alarms as they are received from the network (Fig. 2). *Correlating* alarms means combining the fragmented information they contain and interpreting the flow of alarms as a whole. Alarm correlation systems typically are expert systems performing operations such as (i) *removing* alarms carrying redundant information, (ii) *filtering out* low-priority alarms when higher-priority alarms are present, or (iii) *substituting* a set of alarms by some new information [4]. The goal of alarm correlation is to reduce the amount of information shown to the network managers, improve the usefulness of the information, and ultimately to identify the most probable faults that caused the alarms and to possibly even propose corrective actions.

In addition to the alarms received from a network, several sources of background information are essential in alarm correlation. Knowledge about the topological relationships of senders of alarms is crucial. This contains at least information about the topology of network elements and the managed object containment hierarchy. Information about alarms, e.g., a hierarchy of alarm types, is also useful. A lot of other sources are useful for interpreting alarms. For instance, knowledge about recent problems in the network may help to explain certain alarms.

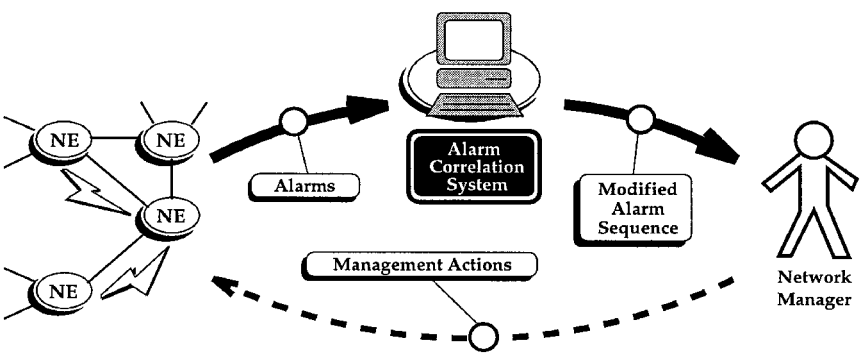

Fig. 2. The flow of alarms from a telecommunication network in an alarm correlation system.

Fig. 3. Example of a correlation action.

We adopt the following formal view to alarm correlation, similar to the one taken by Jakobson and Weissman [3]. Abstractly, the input to a correlation system is an ordered *alarm sequence* $(t_1, s_1, m_1), (t_2, s_2, m_2), \ldots$ of alarm occurrences, where $t_i \leq t_{i+1}$. A *correlation pattern* describes a situation that can be recognized in an alarm sequence within a time window of a given length. Typically, a correlation pattern is an expression on the set of active alarms of, e.g., the last five minutes. If in a given window there is a set of alarms that matches the correlation pattern, then the set is said to be an occurrence of the pattern. Associated with each correlation pattern is a *correlation action*, which is to be executed when there is an occurrence of the corresponding pattern in a window. The correlation action takes care of, e.g., filtering of alarms.

Example 2. Consider a correlation pattern containing two alarms, "*link alarm* from X with severity 1" (alarm type A) and "*high fault rate* in X" (alarm type B), where the variable X may be replaced by the same sender in both alarms. Assume that when alarms of types A and B co-occur, they are known to be followed by fatal problems in their sender X. The correlation action could combine the information in the alarms to a high priority warning message "X will probably collapse within an hour" (message type C), present it to the network manager, and filter out the original alarms (see illustration in Fig. 3).

Given correlation patterns and actions, and a sequence S of alarms, an alarm correlation process continuously observes the incoming alarm sequence S, considers the last time window on S, and executes the actions associated with the patterns that match the window.

Correlation systems need to be prepared to deal with problems caused by delayed alarms, wrong and missing time stamps, and even missing alarms. Small delays are typically no problem, since the alarms can be considered in the order of their time stamps rather than the order of arrival. The correlation system must, then, be capable of backtracking its actions if delayed alarms provide novel information. Unfortunately, there often are inaccuracies in the clocks of network elements and the order of alarms cannot be reliably inferred from the time stamps. Then the correlation system cannot rely on the exact order of the alarms, but, e.g., on the temporal proximity of the alarms. How to handle missing time stamps and missing alarms is very much case dependent. Statistical information about alarms and their occurrences can be useful when deciding how to prepare for missing alarms.

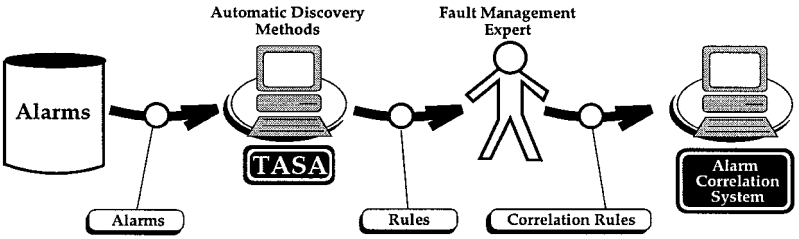

Fig. 4. Use of rules discovered in an alarm database in the construction of alarm correlation systems.

2.3. Building an Alarm Correlation Model

Building a model for alarm correlation is a difficult task. Networks are large and network elements are complex. The number of correlation patterns can be very large, and acquiring them from technical experts is a tedious task.

Correlations may pass unnoticed by the experts, for different reasons. It can be that an expert knows a correlation but did not come to think about it, or a correlation can be such that the expert did not even know there was a connection between certain alarms. Both networks and network elements evolve quickly over time, so a correlation system is never complete. It also takes time for the experts to learn new correlations and to modify existing ones.

We propose semi-automatic methods for the analysis of alarms, to aid in the knowledge acquisition phase, and to give new views to the alarms (Fig. 4). The central idea is to discover recurrent patterns of alarms. Such a pattern can be, for instance, a set of alarms that occurs frequently, or a pattern can state that a certain alarm tends to be followed by another alarm from the same sender.

Briefly, our scenario for building alarm correlation systems is the following.

- First, a large database of alarms is analyzed off-line, and temporal connections between different types of alarms are discovered automatically.
- Then, in the construction of an alarm correlation system, the network management specialists have access to a large collection of alarm patterns and their statistical properties in the analyzed data.
- In the final step, the correlation rules are applied in real-time fault identification.

The construction of alarm correlation systems can hardly be automated: automatically discovered episodes only present statistical properties of recurrent combinations of alarms. Expert knowledge is essential in evaluating the patterns and in assigning proper correlation patterns and actions, not to mention the vast amount of knowledge that cannot be found or expressed using episodes. The purpose of the methods presented in the following section is to help building alarm correlation systems rather than replace them or network management experts.

3. DISCOVERING PATTERNS IN ALARMS

3.1. Overview

TASA discovers two kinds of recurrent patterns, episode rules [24] and association rules [30]. Episode rules describe temporal proximity and temporal ordering of recurrent combinations of alarms in a given alarm database, and they can be used as the basis for correlation patterns. Association rules describe, in turn, the properties of individual alarms without taking the temporal relationships of the alarms into account.

The general form of *episode rules* is the following:

> If a certain combination of alarms occurs within a time period, then another combination of alarms will occur within a time period with a certain probability.

The rule format is fairly simple, but powerful enough to capture various phenomena in alarm sequences. Episode rules are easy to understand, and they closely correspond to the patterns used by many alarm correlation systems.

Example 3. An example of an episode rule is:

> "If alarms of types *link alarm* and *link failure* occur within 5 seconds, then an alarm of type *high fault rate* occurs within 60 seconds with probability 0.7."

Association rules are of the following form:

> If an alarm has certain properties, then it has other given properties, with some confidence.

Most of the useful rules that can be discovered are not 100% certain. TASA discovers rules with arbitrary strengths: it outputs all rules from a class specified by the user, and gives with each rule a confidence factor which tells exactly how strong the rule is in the analyzed alarm database. If there are errors in the analyzed alarms, such as missing alarms, or wrong or missing time stamps, two policies can be adopted. If the errors are known, they can be fixed before episodes are searched for. Alternatively, episodes can be discovered in the original data, in which case the picture given by the discovered episodes can be more realistic. For instance, a confidence close to 100% can suggest that there is an actual causal relation between alarms, but in some cases alarms were missing.

In the rest of this section we describe the rule types in more detail and outline the algorithm that discovers all such rules satisfying certain user-given conditions.

3.2. Episode Rules

The properties of alarms that we consider in the rules are formalized as *alarm predicates*. An alarm predicate φ is an expression that can be evaluated from a single occurrence $a = (t, s, m)$ of an alarm. Alarm predicates typically express properties such as "alarm type is *link failure*", "alarm occurred during office hours", or "alarm type is *link failure* or *link alarm*, and alarm severity class is 3". In episode rules, alarm predicates typically only concern alarm types. With association rules, however, also other types of alarm predicates are useful.

We consider two subtypes of episodes. A serial episode imposes an order on the alarms, whereas an unordered episode ignorant to the order of alarms. Both types of episodes require temporal proximity of the alarms.

A *serial episode* is a sequence $\langle A_1, \ldots, A_k \rangle$ of alarm predicates. Informally, the episode corresponds to k alarms that satisfy the predicates A_i. Formally, given a sequence $S = \langle a_1, \ldots, a_n \rangle$ of alarms, a serial episode $\alpha = \langle A_1, \ldots, A_k \rangle$ *occurs* in S if there is an injective mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, n\}$ such that for all i, $1 \leq i \leq k$, predicate A_i is satisfied by alarm $a_{f(i)}$, and for all i, $1 \leq i \leq k-1$ we have $f(i) < f(i+1)$. We say that an episode $\beta = \langle B_1, \ldots, B_l \rangle$ is a *subepisode* of α if there is an injective mapping $h : \{1, \ldots, l\} \rightarrow \{1, \ldots, k\}$ such that for all i, $1 \leq i \leq l$ we have $A_{h(i)} = B_i$ and for all i, $1 \leq i \leq l-1$ we have $h(i) < h(i+1)$.

Unordered episodes are similar to serial episodes except that the order of the alarms is ignored. An *unordered episode* is a multiset $\alpha = \{A_1, \ldots, A_k\}$ of alarm predicates, and α *occurs* in given sequence $S = \langle a_1, \ldots, a_n \rangle$ if there is an injective mapping $g : \{1, \ldots, k\} \rightarrow \{1, \ldots, n\}$ such that for all i, $1 \leq i \leq k$, predicate A_i is satisfied by alarm $a_{g(i)}$. Another unordered episode β is a *subepisode* of α if and only if $\beta \subseteq \alpha$.

Example 4. Consider the sequence of alarm types in Fig. 5. The serial episode consisting of alarm types E and F in that order occurs several times in the sequence. Alarms E and F may be causally related and perhaps a correlation pattern that predicts F when E has just occurred could be useful—the network management expert should verify this. An unordered episode consisting of alarm types A and B, i.e., A and B in either order, occurs also several times.

Often the events that cause alarms have a certain causal order and, correspondingly, alarms are also emitted in a certain sequence. While serial episodes take the order of alarms into account, unordered episodes are in some cases

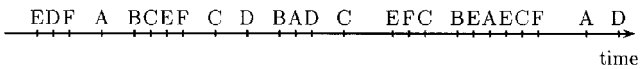

Fig. 5. Example sequence of event types.

more useful in practice: they are more robust with respect to delayed alarms and slightly inaccurate time stamps. Unordered episodes are also more general in the sense that if a serial episode occurs frequently, then the corresponding unordered episode does, too.

The occurrence of both serial and unordered episodes is insensitive to unrelated alarms in the sequence. The alarms of an episode need not be consecutive in the alarm sequence, it is sufficient that correct alarms occur within the time bounds specified. This is obviously useful: it is typical that alarms caused by unrelated problems are merged in the incoming sequence. For the discovery of frequent episodes such unrelated alarms are usually not problematic. If two problems co-occur only by chance, then combinations of alarms from both problems tend to have low frequencies, and episode rules cross-predicting alarms tend to have low confidences.

For an exact formulation of episode rules we consider the concept of a minimal occurrence of an episode. We identify minimal occurrences with the time intervals in which they occur in the following way. Given an episode α and an event sequence S, we say that the interval $[t_s, t_e]$ is a *minimal occurrence* of α in S, (1) if α occurs in the subsequence of S starting at time t_s and ending at time t_e, and (2) if α does not occur in any proper subinterval. The set of minimal occurrences of an episode α in a given event sequence is denoted by $mo(\alpha)$: $mo(\alpha) = \{[t_s, t_e] | [t_s, t_e]$ is a minimal occurrence of $\alpha\}$.

An episode rule gives the conditional probability that a certain combination of alarms occurs within some time bound, given that another combination of alarms has occurred within a time bound. Formally, an *episode rule* is an expression $\beta[win_1] \Rightarrow \alpha[win_2]$, where β and α are episodes such that β is a subepisode of α, and win_1 and win_2 are integers. The interpretation of the rule is that if episode β has a minimal occurrence at $[t_s, t_e]$ with $t_e - t_s \leq win_1$, then the whole superepisode α occurs at interval $[t_s, t'_e]$ for some t'_e such that $t'_e - t_s \leq win_2$.

Each episode rule is characterized by two values. The *confidence* of the rule $\beta[win_1] \Rightarrow \alpha[win_2]$ is the conditional probability (in the analyzed alarm sequence) that α occurs, given that β occurs, under the time constraints specified by the rule. We omit the straightforward but notationally cumbersome exact definition. The *frequency* of an episode rule is the absolute number of times that the whole rule, i.e., α, occurs in the database. Both confidence and frequency are useful measures for the utility of a rule, e.g., in a prediction task: a high confidence means that the rule gives reliable predictions, whereas a high frequency means that the rule can be applied often.

Example 5. An example of episode rule types discovered in the TASA system are:

If an alarm of type *link alarm* is followed by an alarm of type *link failure*

within 5 seconds, then an alarm of type *high fault rate* will follow within 60 seconds with probability 0.7:

$$link\text{–}alarm(a) < link\text{–}failure(b)\ [5] \Longrightarrow\!\!\!\!\!\!< \ high\text{–}fault\text{–}rate(c)\ [60] \qquad (0.7)$$

We use the "less than" operator "<" to denote the order of alarms in serial episodes and episode rules. The corresponding unordered episode rule

$$link\text{–}alarm(a), link\text{–}failure(b)\ [5] \Longrightarrow high\text{–}fault\text{–}rate(c)\ [60] \qquad (0.7)$$

would say that if *link alarm* and *link failure* occur in any order within 5 seconds, then *high fault rate* occurs within 60 seconds, but possibly already between the first two alarms.

These types of rules suggest groups of alarms that are potentially causally related, and also indicate their temporal relation. Given such rules holding in an alarm database, a fault management expert is able to verify whether the rules are useful or not. Some of the rules may reflect known causal connections, some may be irrelevant—and some rules give new insight to the behavior of the network elements.

3.3. Association Rules

Formally, an association rule is an expression $X \Longrightarrow Y$, where X and Y are sets of predicates. Given a set of alarms, the *confidence* of such a rule is the conditional probability with which all the predicates in Y are satisfied by an alarm in the database given that all the predicates in X are satisfied by the alarm. In other words, association rules describe which properties tend to co-occur, and with which probability. The *frequency* of an association rule is the fraction of alarms that satisfy the rule.

Example 6. Examples of association rule types found by TASA are:

- If an alarm a is of type *link alarm*, then it was sent by the network element *EL1* with probability 0.5.

$$link\text{–}alarm(a) \Longrightarrow EL1(a) \qquad (0.5)$$

- If an alarm a is sent during office hours by an element of type base station, then the alarm has severity level 1 with probability 0.9.

$$office\text{–}hours(a), base\text{–}station(a) \Longrightarrow severity\text{–}1(a) \qquad (0.9)$$

Such rules give an overview of what the alarms in the database are like. A

comparison with association rules from last week, last month, etc. can point out trends and changes. Unusually high confidences may point to potential anomalies in the network.

3.4. The Basic Algorithm for Finding Rules

The episode rule discovery task in the TASA system can be formalized as follows. Given a sequence S of alarms, a set E of alarm predicates, a frequency threshold c, and a set W of integers, find the confidences of all episode rules of the form $\beta[win_1] \Rightarrow \alpha[win_2]$, where β and α are episodes built from alarm predicates in E, β is a subepisode of α, and win_1, $win_2 \in W$, and whose frequency is at least c. With different specifications of these parameters, of E in particular, the user can have different views to the analyzed data. We return to the issue of specifying the parameters in Section 4.1.

For association rule discovery, an alarm database S, a set E of alarm predicates, and a frequency threshold c are given, and the task is to find the confidences of all rules $X \Rightarrow Y$, where $X, Y \subseteq E$, whose frequency is at least c.

The main algorithms for the two tasks are based on the same basic idea. Our focus will be on the more complex task of discovering episode rules. We first discuss how the frequently occurring episodes (or predicate sets for association rules) are computed. We then show how rules can be obtained from this information.

3.4.1. Computing Frequent Episodes or Predicate Sets

The problem we consider is the following. Given S, E, c, and W as before, find all episodes that have at least c minimal occurrences of length at most win^*, where $win^* = \max\{w \in W\}$. The following algorithm outputs all such episodes.

1. $C_1 := \{\{e\} \mid e \in E\}$;
2. $i := 1$;
3. **while** $C_i \neq \emptyset$ **do**
4. *recognition*: read the sequence S, and let F_i be the collection of episodes of C_i that occur at least c times;
5. *building*: compute C_{i+1} to contain those episodes that have $i + 1$ alarm predicates and whose all subepisodes occur at least c times;
6. $i := i + 1$;
7. **od**;
8. **for** all i, output F_i;

The algorithm starts with simple episodes and proceeds to larger episodes. It works iteratively: first a candidate collection C_i is generated, candidates are counted in the alarm sequence, those occurring often enough are saved in the collection F_i, and finally new candidates are generated again.

Essential for the efficiency of the algorithm is the following observation.

If an episode does not occur often enough, then its superepisodes—which are more specific—cannot occur often enough. Therefore, the candidate collection C_i of episodes is built (Step 5) to contain only episodes whose *all* subepisodes occur often enough. More details can be found in [24].

To find frequent predicate sets, essentially the same algorithm and the same observation hold. Candidates are now sets of predicates, and the input S is seen as an unordered collection of alarms. In Step 4, "recognition", for each candidate set the number of satisfying alarms is computed and compared to the frequency threshold. In Step 5, "building", new candidate sets are constructed such that *every* subset of a candidate set is frequent [31].

This basic algorithm can be modified to take into account, e.g., the network topology, the types of network elements, or an alarm type hierarchy. For instance, episodes can be required to consist of alarms from network elements whose distance is small in the network topology. A simple way to implement such a restriction is to modify Step 4, the recognition of episodes: only accept occurrences such that the alarms fulfill the desired restriction. The properties of different types of networks and network elements can be used to design more specialized knowledge discovery methods. For reasons of brevity, we restrict ourselves to this earlier algorithm without such modifications.

3.4.2. Rule Generation

We now show how the confidences of rules can be computed, once the frequent episodes or predicate sets are known.

Recall that we defined an episode rule as an expression $\beta[win_1] \Longrightarrow \alpha[win_2]$, where β and α are episodes such that β is a subepisode of α, and win_1 and win_2 are integers. To find such rules, first note that for the rule to be frequent, the episode α has to be frequent. So rules of this form can be enumerated by looking at all frequent episodes α, and then looking at all subepisodes β of α. The evaluation of the confidence of the rule $\beta[win_1] \Longrightarrow \alpha[win_2]$ can be done in one pass through the minimal occurrences of β and α, as follows. For each minimal occurrence $[t_s, t_e]$ of β with $t_e - t_s \leq win_1$, locate the first minimal occurrence $[u_s, u_e]$ of α such that $t_s \leq u_s$. Then check whether $u_e - t_s \leq win_2$.

For association rules the rule generation is even simpler. Given a frequent predicate set X, for all its subsets $Y \subset X$ we have that the confidence of the rule $(X/Y) \Longrightarrow Y$ is the frequency of X divided by the frequency of X/Y. Since the frequencies of all frequent sets have already been computed in Step 4, the database is not needed for the rule generation at all.

4. RULE DISCOVERY PROCESS WITH TASA

Our discovery methods find *all* rules that hold in the given alarm database with respect to user-given specifications. The idea is that a large collection of

valid patterns is discovered at once, and different views to the data are then supported effectively by different views to the collection of patterns. The discovery step is fully automatic, given the parameters of the algorithm. The second part, viewing the episodes, contains a minimal amount of computation and is based on user interaction only.

In this section we review two important aspects of this approach: specification of the criteria according to which all rules are automatically discovered, and the user interface methodology for browsing large collections of rules. These two steps are where the network management expertise is brought into the process of discovering patterns.

4.1. Rule Discovery

The user specifies the following parameters for the discovery of rules (Fig. 6).

4.1.1. Alarm Predicates

Alarm predicates are the expressions used to refer to the (properties of) alarms, and they are given by the user. For episode rules, the type of the alarm and the sender of the alarm are the most typical predicates.

In the most simple case only the alarm type is considered. Then episodes reveal connections between types of alarms without respect to the network elements that sent the alarms. Alternatively, e.g., predicates specifying (sender,

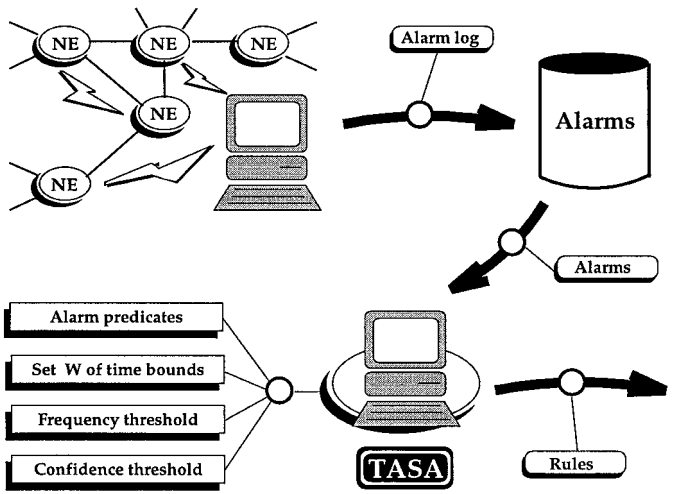

Fig. 6. The environment of the TASA system.

alarm type) pairs can be considered, making it explicit that the input is merged from alarms from several senders in the network. An episode rule found with predicates like this shows connections between different types of alarms from particular network elements. Predicates consisting of the (sender type, alarm type) pair have actually proved to be one of the most useful forms: episode rules between types of alarms in different types of devices seem to describe the network behavior in a reasonable level of abstraction.

For association rules we consider also predicates such as the priority of the alarm, the day of the week, whether the alarm occurred during office hours or not, etc.

4.1.2. Set of Time Bounds

For episode rules, the user also supplies a set W of time bounds with which the rules are constructed. Two aspects guide the setting of W. (1) The maximum time bound in W should be larger than the maximal temporal duration of the phenomena that are searched for. (2) The number of time bounds in W directly affects the temporal granularity at which episode rules are found. Our fault management experts have typically preferred time bounds ranging from 5 seconds to 10 minutes, e.g., with roughly logarithmically growing time bounds 5 s, 10 s, 30 s, 1 min, 2 min, 5 min, 10 min. The higher the number of time bounds is, the higher the number of rules is, too. The effect on the running time is not strong.

4.1.3. Frequency Threshold

For both episode and association rules, a *frequency threshold* is given by the user. The method outputs all episode and association rules specified by the earlier parameters, *whose frequency is at least the user-specified threshold.*

Recall that the method is aimed at discovering statistical rules, not spotting interesting individual cases. With the frequency threshold the user is able to filter out rules that are too rare to be trustworthy. For instance, with a frequency threshold of 20 an episode is output only if it appears at least 20 times in the analyzed database. The algorithm is complete in this respect: it is guaranteed to output all episodes that have at least 20 occurrences.

The frequency threshold is crucial for the running time of the algorithm. If the threshold is low, also rules that occur rarely are included in the output, and the discovery time is longer. Suitable values depend heavily on the nature and amount of data. For a database with 100 000 alarms, thresholds in the range of 50 to 500 may be reasonable.

4.1.4. Confidence Threshold

For both episode and association rules, the user also specifies a *confidence threshold* d. The algorithm then outputs all episode and association rules *whose confidence is at least* d.

The confidence threshold allows the user to filter out rules that are too weak

to be useful. For instance, a confidence threshold of 80% limits the output to rules that hold with at least 80% certainty. The confidence threshold has no particular effect on the running time, so it is useful to specify a low threshold in the rule discovery phase and to prune weak rules later interactively with the user interface tools.

The discovery method outputs each episode rule and association rule satisfying the above conditions. The conditions should typically be quite loose, so large amounts of rules are discovered. For each rule, TASA outputs the confidence and the frequency, and an estimate of the statistical significance of the rule.

4.2. User Interface to Discovered Rules

Our motivation for discovering a lot of rules at once is that network management expert's requests for different viewpoints to the data can then be responded very quickly: a new pattern discovery phase is not necessary, but simply a new view to the already discovered patterns.

TASA offers a variety of focusing and ordering criteria for rules and supports iterative retrieval from the discovered knowledge. Network management experts can manipulate the rule set using selection and sorting operations, as well as more complex operations for including or excluding certain classes of rules. The following types of operations are supported for creating and manipulating views to the collection of discovered rules.

1. *Focusing*: presentation only of a subset of rules, after explicit removals of uninteresting rules and selections of (potentially) interesting ones.
2. *Sorting* of rules according to various criteria.
3. *Clustering*: grouping of rules into classes of related rules.

While creating a focus, simple threshold-like restrictions, such as rule frequency and confidence may satisfy a large number of rules. In our approach, this problem can be alleviated by selecting rules to or removing rules from the view by *templates* [32]. Hoschka and Klösgen [14] have also used templates for defining interesting knowledge, and their ideas have strongly influenced our work. Their approach is based on few fixed statement types and partial ordering of attributes, whereas our approach is closer to regular expressions.

We define templates as simple regular expressions that describe, in terms of alarm predicates, the form of rules that are to be shown or not shown. More formally, a template is an expression

$$A_1, \ldots, A_k \Longrightarrow A_{k+1}, \ldots, A_l,$$

where each A_i is either an alarm predicate, the name of an alarm predicate collection, or an expression $C+$ or C^*, where C is a collection name.[5] Here $C+$ and C^* correspond to one or more and zero or more instances of the collection C, respectively. A rule

$$B_1, \ldots, B_m \Rightarrow B_{m+1}, \ldots, B_n$$

matches a template if the rule can be considered to be an instance of the pattern. This simple technique, sketched in Example 7, is surprisingly powerful.

Example 7. Focus can be set to, e.g., day-time alarms by selecting only association rules that contain the predicate "office hours = yes". Or, episode rules containing alarms from separate subnetworks can be obtained by using templates that reject all rules where the senders are in the same subnetwork.

The template concept can be combined with thresholds for rule frequency, confidence, and significance. The user may state restrictions such as "rule frequency must be between 5% and 30%", "rule confidence must be at least 80%", and "rule significance must be over 0.95". In this case the user filters out very rare and reasonably frequent rules, and further on selects only those that are both strong and statistically significant.

Several selecting and removing templates can be used simultaneously to achieve the desired viewpoint. To be shown, a rule must match all positive templates and none of the negative ones.

An essential feature of TASA is the support for iterative browsing of rules: templates can be added and edited easily, so it is convenient to try out different focuses to the data. This is very useful for exploratory analysis of the alarm data. Typically the most useful findings are unexpected, and templates allow flexible navigation in the data when tracing hints for interesting rules.

Sorting of the rule set can be based on the alarm predicates in the rules, e.g., types or severities of the alarms, the sender, etc., or on the numerical values of the confidence, frequency, or statistical significance of the rules. For instance, in Example 7 the resulting rule set could be arranged so that the rules are listed in descending order by their significance and confidence, or in ascending alphabetical order. Although sorting can be considered as rather trivial operation, it is often enough to fulfill basic clustering needs.

Clustering of rules aims at giving a larger picture of the behavior of the alarm sequence. In the data there are often various explanations for the occurrence of a particular alarm type, say *path unavailable*. Clustering methods can be used to assign rules to groups so that two rules with the right-hand side *path unavailable* belong to the same cluster if they often explain or predict *path*

[5] A_i can also be a regular expression.

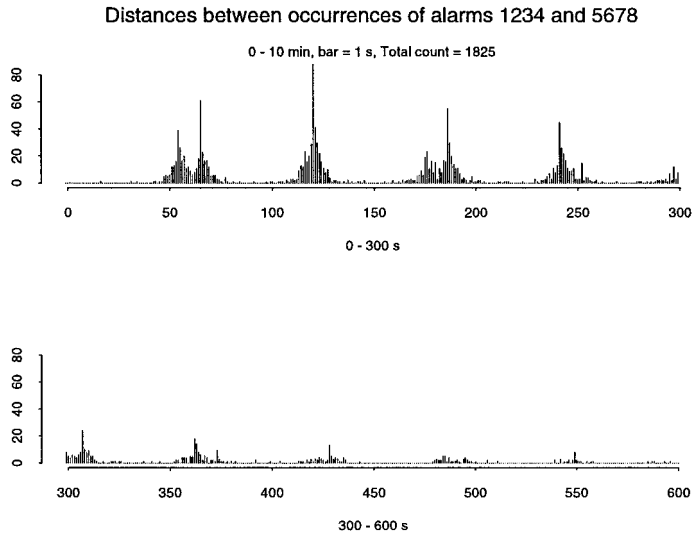

Fig. 7. Distance histogram for the left-hand and right-hand sides of a rule 1234 $\Longrightarrow$5678. The x-axis represents time in seconds, the y-axis the number of occurrences when alarm 1234 is followed by 5678 after x seconds.

unavailable in similar situations [33]. This can be useful in pointing out potentially related rules. By extending the clustering to a Scatter/Gather like method [34], clustering results could be utilized interactively in browsing the rule set.

For visualization the TASA system offers some simple facilities. For example, an interesting picture of the interaction between the left and right-hand sides of an episode rule can be obtained by drawing a histogram showing the distance from each occurrence of the left-hand side to the nearest occurrence of the right-hand side. Such histograms are valuable guides for locating possible periodic relationships between the left and right-hand sides, as is demonstrated by Fig. 7. Note that the episode rule formalism does not fully capture the details of such relationships.

The original user interface of TASA was based on HTML language to be used with standard WWW browsers for HTML documents. The current version of TASA, JTASA, is written in Java language and it is a client/server system. The main component of the system is the JTASA server, which takes care of the management of original data as well as the generation and management of the results produced according to the users' requests. Thus, JTASA clients only initiate actions at the server side and then show the results.

Briefly, the structure of the system is as follows. At client side, there is a file manager window from which the user can both select data sets for association

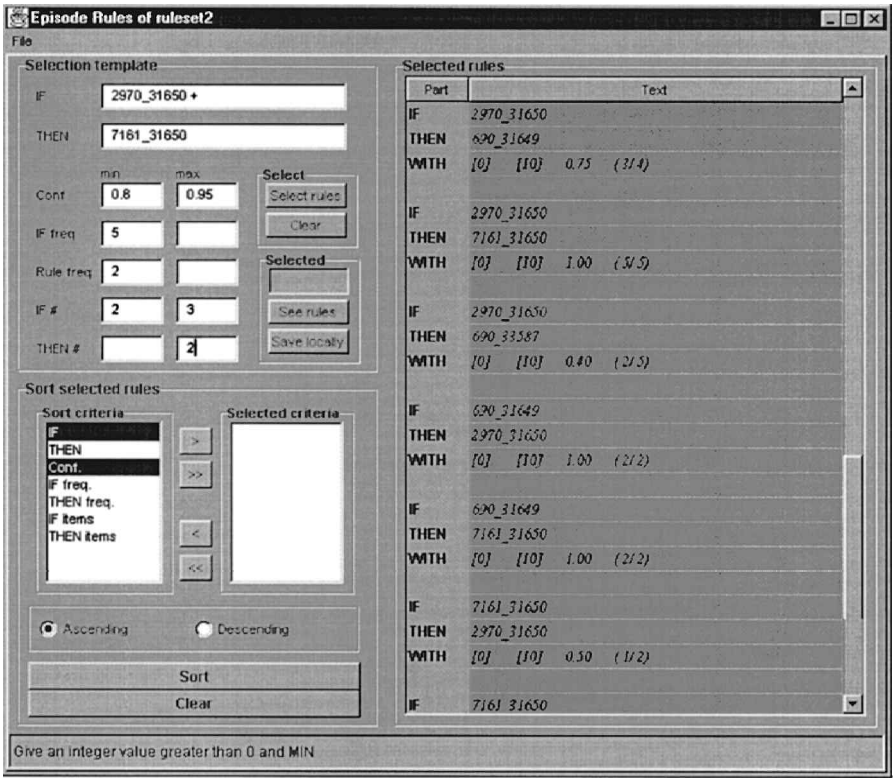

Fig. 8. Example window of the new TASA system (rule viewing).

and episode rule creation and view generated rule sets. In the rule creation part, the user can give detailed instructions for rule generation as described in Section 4.1. The rule viewing window (Fig. 8) is based on the use of templates with some additional features for defining the number of items in IF and THEN parts of the rule, and for sorting the rules. Other functionalities include statistics about the data set, alarms of the data set, and predicates used.

5. EXPERIENCES IN USING TASA

Different versions of TASA have been in prototype use in four telecommunication companies since the beginning of 1995, and experiences are encouraging. In Section 5.1 we discuss hands-on experience in using TASA and give examples of the use of discovered rules. We report on performance evaluations in Section 5.2.

5.1. TASA in Practice

The fault management experts in the telecommunication companies have found TASA useful in

- finding long-term, rather frequently occurring dependencies,
- creating an overview of a short-term alarm sequence, and
- evaluating the alarm data base consistency and correctness.

Unexpected dependencies have been found, e.g., between network elements which are not closely connected in the network topology. An example of such a dependency is that when a remote device sends alarms, the fault is reflected to another corner of the network through several devices, and not always necessarily via the same routes and devices. So, just analyzing the neighboring devices might not reveal any strong relationships. However, when a larger region is analyzed, such a relationship can be detected. Beginning from the first tests, discovered rules have been integrated into alarm correlation systems.

On the other hand, many of the rules discovered by TASA are deemed trivial by the network managers. Some of the rules correspond to the knowledge that the network managers have about the behavior of the network, and some other rules reflect the assumed functioning of network devices. An example of the former is, for instance, that it is known that if an alarm of type A occurs, it is always followed by an alarm of type B within 20 seconds; so, a found rule like this gives no new information. Luckily, much of the trivial knowledge can be expressed and removed with templates. Templates are also useful since the knowledge trivial to one expert may not be trivial to another, and with templates each expert may filter the rule collection based on his/her personal background knowledge.

The experiences have indicated that the algorithms are not well suited for analyzing event sequences that contain long bursty periods. It is sometimes more useful to cut off and analyze such periods separately.

The usability of discovery tools has an essential, often perhaps under-estimated role. The usability of an early version of TASA was tested in the usability laboratory of the Helsinki University of Technology. The tests contained, e.g., user tests taken by four fault management experts from telecommunication companies. In the tests, TASA was acknowledged both visually and environmentally appealing. On the other hand, first-time users were unfamiliar with many concepts from the knowledge discovery field. Despite these problems with the terminology, the system as a whole got encouraging comments.

Overall, TASA has been considered useful. Episode rules are being used as first drafts of correlation rules, whereas association rules are more typically used for creating short-term overviews in off-line network surveillance. Telecommunication operators are integrating these methods to their alarm analysis and surveillance systems. Next we give examples of the use of discovered rules.

5.1.1. Knowledge Acquisition for Alarm Correlation

We use as an example an alarm correlation system which operates in real time and is also able to handle delayed alarms and slightly inaccurate time stamps. In the correlation patterns, delays are handled with a special *wait* function. Episode and association rules can be applied in this system in a rather straightforward way. The rule

$$link\ \ alarm < link\ \ failure\ [5] \Longrightarrow\!\!\!\!\!\!\times\ high\ \ fault\ \ rate \qquad [60] \qquad (0.7)$$

discovered by TASA can be coded in the system as follows:

```
if "alarm type = link alarm" then
    start time;
    wait until "alarm type = link failure" or "time = 5 s";
    if "alarm type = link failure" then
        send an alarm "high fault rate with 70% probability in 60 s"
    else forward the original alarm;
    fi;
fi
```

That is, if a link alarm occurs and a link failure follows within 5 seconds, the rule right-hand side information is sent and the original alarms are suppressed. If a link failure does not follow within 5 seconds, the original link alarm is forwarded.

The correlation procedure can be enhanced using association rules. For example, assume that we have detected that if the alarm type is *link alarm*, the time of the day is office hours, and alarm severity is 1, then with probability of 95% the alarming element type is *BS*. After expert evaluation of the rule we know that such alarms from network elements of type *BS* are of no interest. However, if an element of any other type sends that alarm, it is an interesting one. Thus, we can specify the following correlation function that removes such alarms.

```
if "alarm type = link alarm" and "office hours" and "severity = 1" and
    "element type = BS" then
    exit
else forward the original alarm;
fi
```

5.1.2. Network Surveillance

As an example of how the system can be used for off-line network surveillance, consider the following typical scenario. Assume the network manager has used TASA to discover association rules for the current month. First he might want to see what the alarms have been like during the current week, say week 30, so he uses a template to select rules with the predicate "week = 30" as the left-hand side.

The number of selected rules is still very large. The network manager decides to restrict the rule right-hand side to only contain predicate, and he also sorts the rules by their confidences.

Looking at the selected rules, he sees the rule "if week = 30 then alarm type = *connection failure*" with confidence 0.12, and he infers that an unusually large fraction of alarms during the week has been of type *connection failure*. To see in more detail what the alarms have been like, he refines the template and selects rules with "week = 30 and alarm type = *connection failure*" as the left-hand side.

Looking at the new set of selected rules, the network manager sees that a lot of rules concern the network element *EL1*. That reminds him of maintenance undertaken in the beginning of the week that explains those rules. To remove the rules, he applies a rejective template with the predicate "network element = *EL1*".

The resulting set of rules shows nothing special, but just to make sure the network manager wants to compare the rules with the corresponding rules from some previous week. He opens a copy of the window, and changes the first template to "week = 29". If there is anything special or interesting, the viewing criteria can be refined or altered again.

5.1.3. Rules in Changing Environments

Telecommunication networks usually change and evolve quickly over time. It should be noted that even small changes can sometimes affect the behavior of the network substantially. In practice, this means that old rules that hold at a certain moment do not necessarily hold in a changing environment, whether because of actual topological changes in the devices, software etc. The strength of our approach is that reruns to get valid rules can be done efficiently. However, human evaluation is needed in deciding whether the rerun requires modifications to the rule-base of the correlation expert system.

5.2. Performance Results

We have evaluated the efficiency of the episode discovery algorithm using several alarm databases from fixed and cellular networks. Typical running times on a Pentium-based PC range from few seconds to an hour, depending on the database and the parameters. Episodes with the alarm type as the only predicate can be discovered in a sequence of 70 000 alarms with a window width of 60 seconds in few seconds, whereas finding association rules with over 8 000 different predicates can take an hour (the 8 000 predicates were derived by considering about every bit of information contained in alarm messages, e.g., "seconds = 0", "seconds = 1", etc).

The following representative results have been obtained with an alarm

Table I. Performance for Serial Episodes[a]

Frequency threshold	Candidates	Frequent episodes	Iterations	Total time (s)
50	12732	2735	83	28
100	5893	826	71	16
250	2140	298	54	16
500	813	138	49	14
1000	589	92	48	14
2000	405	64	47	13

[a]Maximum time bound $win^* = 60$ s.

database consisting of 73 679 alarms and covering a time period of 7 weeks. We examined the sequence of alarm types; there were 287 different types with diverse frequencies and distributions. The experiments have been run on a PC with 166 MHz Pentium processor and 32 MB main memory, under the Linux operating system. A flat text file copy of the necessary alarm information was used in the tests.

Tables I and II represent performance statistics for finding frequent episodes in the alarm database with various frequency thresholds. The number of frequent episodes decreases rapidly as the frequency threshold increases. The running times are moderate, between 12 and 30 seconds for thousands of discovered episodes. The number of iterations equals the size of the largest episodes considered.

The method can easily produce very large amounts of rules. Table III represents experimental rule generation results with serial episodes and with a varying number of possible time bounds in W. The rules have been obtained with maximum time bound 60 s and with confidence threshold 0. The time requirement increases slowly as more time bounds are used, but the time increases slowlier than the number of rules. The initial episode generation took around 14 s.

Table II. Performance for Unordered Episodes[a]

Frequency threshold	Candidates	Frequent episodes	Iterations	Total time (s)
50	10041	4856	89	30
100	4376	1755	71	20
250	1599	484	54	14
500	633	138	49	13
1000	480	89	48	12
2000	378	66	47	12

[a]Maximum time bound $win^* = 60$ s.

Table III. Experimental Rule Generation Results with Serial Episodes[a]

Number of time bounds	Rules	Rule gen. time (s)
1	1221	13
2	2488	13
4	5250	15
10	11808	18
20	28136	22
30	42228	27
60	79055	43

[a]Maximum time bound $win^* = 60$ s, confidence threshold 0, frequency threshold 1000.

The amount of almost 80 000 rules, obtained with 60 time bounds, may seem unnecessarily large. There are, however, only 1221 distinct rules if the time bounds are ignored; the rest of the rules present different combinations of time bounds, in this case down to the granularity of one second. For the cost of 43 s we thus obtain very fine-grained rules from our frequent episodes. Different criteria can then be used to select the most interesting rules from these. Figure 9 represents the effect of the confidence threshold to the number of distinct rules found. Although the initial number of rules may be quite large, it decreases fairly rapidly if we require a reasonable confidence.

Example 8. We have analyzed several real-world datasets from telecommunication companies. To this example, we have selected four datasets. These datasets, described in more detail in Table IV, have partly been preprocessed so that only actual alarms have been selected, if also warnings, notifications etc. have appeared in the data sequence. That is, the original data sequence contains usually substantially more items without preprocessing.

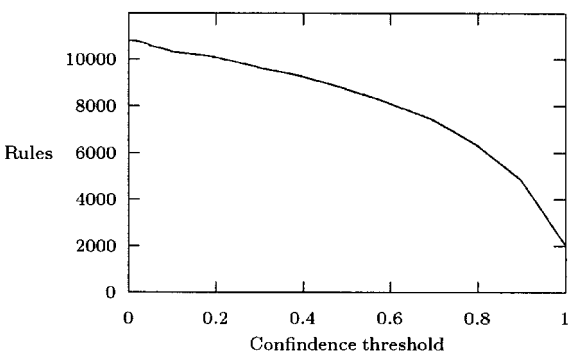

Fig. 9. Total number of distinct rules found with various confidence thresholds; maximum time bound 60 s, frequency threshold 100.

The datasets represent typical usage of TASA system in analysing both short-term sequences, a couple of days, and long-term sequences, a couple of months. They also reflect the real-world situation, where the material to be analysed contains plenty of different types of events, i.e., here hundreds or thousands of types of alarms. In the resulting rules, however, many alarm types are not present at all due to their low frequencies that do not exceed the given thresholds; see Table V.

As can be seen in Table V, the number of resulting rules can be quite large even when the threshold values have been properly selected. This is not a problem, however, because the tool described in Section 4 can be applied to browse the rule collection efficiently.

The time requirement of the algorithm is linear in the number of alarms, and much larger databases can be analyzed with acceptable response times.

6. CONCLUSIONS

Fault management is a critical but difficult task in network management. The flow of alarms received by a management center should be correlated automatically to a more intelligible form, in order to facilitate identification and correction of faults. Unfortunately, the construction of an alarm correlation system requires a lot of both expertise and time, and is a process that never is complete.

We proposed automatic methods for the discovery of recurrent patterns in alarm databases, in order to help in the construction of alarm correlation systems. We described a methodology for the discovery of episode rules and association rules in telecommunication alarm databases. Episode rules describe temporally related sets of alarms, and they are useful for building and augmenting alarm correlation systems. Episode rules also have potential for anomaly detection. Association rules describe statistical properties of single alarms without taking temporal relationships into account. The discovery of association rules has proved to be valuable in network surveillance.

A prototype system, TASA, implements these ideas. Using TASA, telecommunication operators have been able to discover new and useful regularities for their alarm correlation systems. The discovery methods are also being integrated to the alarm analysis and surveillance systems of operators.

ACKNOWLEDGMENTS

The authors would like to thank Dr. A. Inkeri Verkamo for providing experimental results. Valuable comments from anonymous reviewers are gratefully acknowledged.

This work has been supported by the Academy of Finland, National Technology Agency (TEKES), and Nokia Foundation.

Table IV. Example Dataset Characteristics and Alarm Occurrence Times

Dataset 1		Dataset 2	
First alarm	16.08.96 at 21:32:33	First alarm	13.10.96 at 00:00:20
Last alarm	19.08.96 at 04:08:32	Last alarm	25.10.96 at 11:46:54
Duration	2d:6h:36m:0s	Duration	2d:6h:36m:0s
Number of alarms	19 429	Number of alarms	58 616
Number of alarm types	1 982	Number of alarm types	1 956
Frequency threshold	10	Frequency threshold	25
Confidence threshold	0.1	Confidence threshold	0.2
Window sizes	5,10,20,30,60	Window sizes	5,10,20,30,60
Number of distinct rules	9 321	Number of distinct rules	778
Dataset 3		**Dataset 4**	
First alarm	12.10.96 at 21:35:14	First alarm	01.04.96 at 07:38:35
Last alarm	21.10.96 at 16:57:04	Last alarm	10.12.96 at 14:29:52
Duration	8d:19h:21m:50s	Duration	253d:6h:51m:17s
Number of alarms	22 158	Number of alarms	98 413
Number of alarm types	41	Number of alarm types	212
Frequency threshold	10	Frequency threshold	50
Confidence threshold	0.05	Confidence threshold	0.2
Window sizes	5,10,20,30,60	Window sizes	5,10,20,30,60
Number of distinct rules	387	Number of distinct rules	9 406

Table V. Episodes and Rules Exceeding the Thresholds in Example Datasets

Dataset	Singleton episodes	Total No. of episodes	Max episode size	Rules
1	242	4112	10	9 321
2	299	1318	4	1 318
3	18	325	4	325
4	58	7438	6	7 438

REFERENCES

1. J. Goldman, P. Hong, C. Jeromnimon, G. Louit, J. Min, and p. Sen, Integrated fault management in interconnected networks. In B. Meandzija and J. Westcott (eds.), *Integrated Network Management, I*, Elsevier, pp. 333–344, 1989.
2. R. M. Goodman, B. E. Ambrose, H. W. Latin, and C. T. Ulmer, Noaa—An expert system managing the telephone network, In *Integrated Network Management IV*, Chapman and Hall, London, pp. 316–327, 1995.
3. G. Jakobson and M. Weissman, Real-time telecommunicatio n network management: Extending event correlation with temporal constraints, *Integrated Network Management IV*, Chapman and Hall, London, pp. 290–301, 1995.
4. G. Jakobson and M. D. Weissman, Alarm correlation, *IEEE Network*, Vol. 7, No. 6, pp. 52–59, 1993.
5. J. F. Jordaan and M. E. Paterok, Event correlation in heterogenous networks using OSI management framework, In H.-G. Hegering and Y. Yemini (eds.), *Integrated Network Management, III*, Elsevier Science Publishers B.V (North Holland), Amsterdam, The Netherlands, pp. 683–695, 1993.
6. Y. A. Nygate, Event correlation using rule and object based techniques, *Integrated Network Management IV*, Chapman and Hall, London, pp. 278–289, 1995.
7. A. T. Bouloutas, S. B. Calo, A. Finkel, and I. Katzela, Distributed fault identification in telecommunication networks, *Journal of Network and Systems Management*, Vol. 3, No. 3, pp. 295–312, 1995.
8. M. Trommer and R. Konopka, Distributed network management with dynamic rule-based managing agents, *Integrated Network Management V*, Chapman and Hall, London, pp. 730–741, 1997.
9. R. Milne, C. Nicol, M. Ghallab, L. Trave-Massuyes, K. Bousson, C. Dousson, J. Quevedo, J. Aguilar, and A. Guasch, TIGER: Real-time situation assessment of dynamic systems, *Intelligent Systems Engineering*, pp. 103–124, 1994.
10. D. Medhi and D. Tipper (eds.), Special issue: Fault management in communication networks, *Journal of Network and Systems Management*, Vol. 5, No. 4, 1997.
11. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, California 1996.
12. G. Piatetsky-Shapiro and W. J. Frawley (eds.), *Knowledge Discovery in Databases*, AAAI Press, Menlo Park, California.
13. R. M. Goodman and H. Latin, Automated knowledge acquisition from network management databases, In I. Krishnan and W. Zimmer (eds.), *Integrated Network Management, II*, Elsevier Science Publishers B.V (North-Holland), Amsterdam, The Netherlands, pp. 541–549, 1991.
14. P. Hoschka and W. Klösgen, A support system for interpreting statistical data, In G. Piatetsky-

Shapiro and W. J. Frawley (eds.), *Knowledge Discovery in Databases*, AAAI Press, Menlo Park, California pp. 325–345, 1991.
15. W. Kloesgen, Efficient discovery of interesting statements in databases, *Journal of Intelligent Information Systems*, Vol. 4, No. 1, pp. 53–69, 1995.
16. R. Zembowicz and J. M. Zytkow, From contingency tables to various forms of knowledge in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, California pp. 329–349, 1996.
17. C. J. Matheus, G. Piatetsky-Shapiro, and D. McNeill, Selecting and reporting what is interesting. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, California pp. 495–515, 1996.
18. G. Piatetsky-Shapiro and C. J. Matheus, The interestingness of deviations. In U. M. Fayyad and R. Uthurusamy (eds.), *Knowledge Discovery in Databases, Papers from AAAI Workshop (KDD'94)*, Seattle, Washington, July 1994, pp. 25–36.
19. R. Agrawal and R. Srikant, Mining sequential patterns, *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, Taipei, Taiwan, pp. 3–14, March 1995.
20. C. Bettini, X. S. Wang, and S. Jajodia, Testing complex temporal relationships involving multiple granularities and its application to data mining, *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, Montreal, Canada, June 1996, pp. 68–78.
21. A. E. Howe, Finding dependencies in event streams using local search, *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, pp. 271–277, January 1995.
22. I. Jonassen, J. F. Collins, and D. G. Higgins, Finding flexible patterns in unaligned protein sequences, *Protein Science*, Vol. 4, No. 8, pp. 1587–1595, 1995.
23. P. Laird, Identifying and using patterns in sequential data. In K. P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori (eds.), *Algorithmic Learning Theory, Fourth International Workshop*, Berlin, pp. 1–18, 1993. Springer-Verlag.
24. H. Mannila, H. Toivonen, and A. I. Verkamo, Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 259–289, 1997.
25. T. Oates and P. R. Cohen, Searching for structure in multiple streams of data, *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, Morgan Kaufmann, San Francisco, California, pp. 346–354, July 1996.
26. J. T.-L. Wang, G.-W. Chirn, T. G. Marr, B. Shapiro, D. Shasha, and K. Zhang, Combinatorial pattern discovery for scientific data: Some preliminary results. In R. T. Snodgrass and M. Winslett (eds.), *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'94)*, ACM, Minneapolis, Minnesota, pp. 115–125, June 1994.
27. C. L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, *Artificial Intelligence*, Vol. 19, pp. 17–37, 1982.
28. R. Grossi and F. Luccio, Simple and efficient string matching with k mismatches, *Information Processing Letters*, Vol. 33, pp. 113–120, 1989.
29. G. Das, R. Fleischer, L. Gasieniec, D. Gunopulos, and J. Kärkkäinen, Episode matching, *Proceedings of the Eigth Annual Symposium on Combinatorial Pattern Matching (CPM'97)*, Aarhus, Denmark, pp. 12–27, June 1997.
30. R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia (eds.), *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, ACM, Washington, D.C., pp. 207–216, May 1993.
31. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances*

in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, California, pp. 307–328, 1996.
32. M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo, Finding interesting rules from large sets of discovered association rules, *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, ACM, Gaithersburg, Maryland, pp. 401–407, November 1994.
33. H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hätönen, and H. Mannila, Pruning and grouping of discovered association rules, *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, MLnet, Heraklion, Crete, Greece, pp. 47–52, April 1995.
34. D. R. Cutting, D. Karger, J. Pedersen, and J. W. Tukey, Scatter/gather: A cluster-based approach to browsing large document collections, *Proceedings of the 15th Annual International ACM/SIGIR Conference*, Copenhagen, Denmark, pp. 318–329, June 1992.

Mika Klemettinen is an assistant professor of Computer Science at the University of Helsinki, Finland. He has been working in the Data Mining Group since Fall 1994. He obtained his Ph.D. in Computer Science from the University of Helsinki in 1999, with a thesis titled "A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases." Between 1994 and 1997 he worked as a project manager in a Data Mining project funded by the National Technology Agency (TEKES) and four Finnish telecommunication companies. In this project he participated in developing the TASA system and data mining methodologies. His main research interests include data mining, text mining, and information retrieval.

Heikki Mannila is a professor of Computer Science at the University of Helsinki, where he also obtained his Ph.D. in 1985. After that he has been an associate professor at the Universities of Tampere and Helsinki, a visiting professor at the Technical University of Vienna, and a guest researcher at the Max Planck Institute für Informatik in Saarbrücken. He also worked at the National Public Health Institution in Helsinki, as well as a consultant in industry. Mannila has been a member of numerous program committees and he was one of the program chairman of KDD-97. He is also one of the editors-in-chief of the journal *Knowledge Discovery and Data Mining* published by Kluwer Academic Publishers. His research interests include rule discovery from large databases, the use of Markov chain Monte Carlo techniques in data analysis, and the theory of data mining.

Hannu Toivonen is an assistant professor of Computer Science at the University of Helsinki. Prior to joining the university, he was a researcher engineer at Nokia Research Center, where he was involved with knowledge-based systems and methods for telecommunication network management. Hannu Toivonen earned his Ph.D. in Computer Science from the University of Helsinki in 1996 on data mining, with a thesis titled "Discovery of frequent patterns in large data collections." He is one of the developers of the TASA knowledge discovery system and the implementor of the data mining algorithms. His current research interests are in data mining and in the use of Markov chain Monte Carlo methods for data analyses.